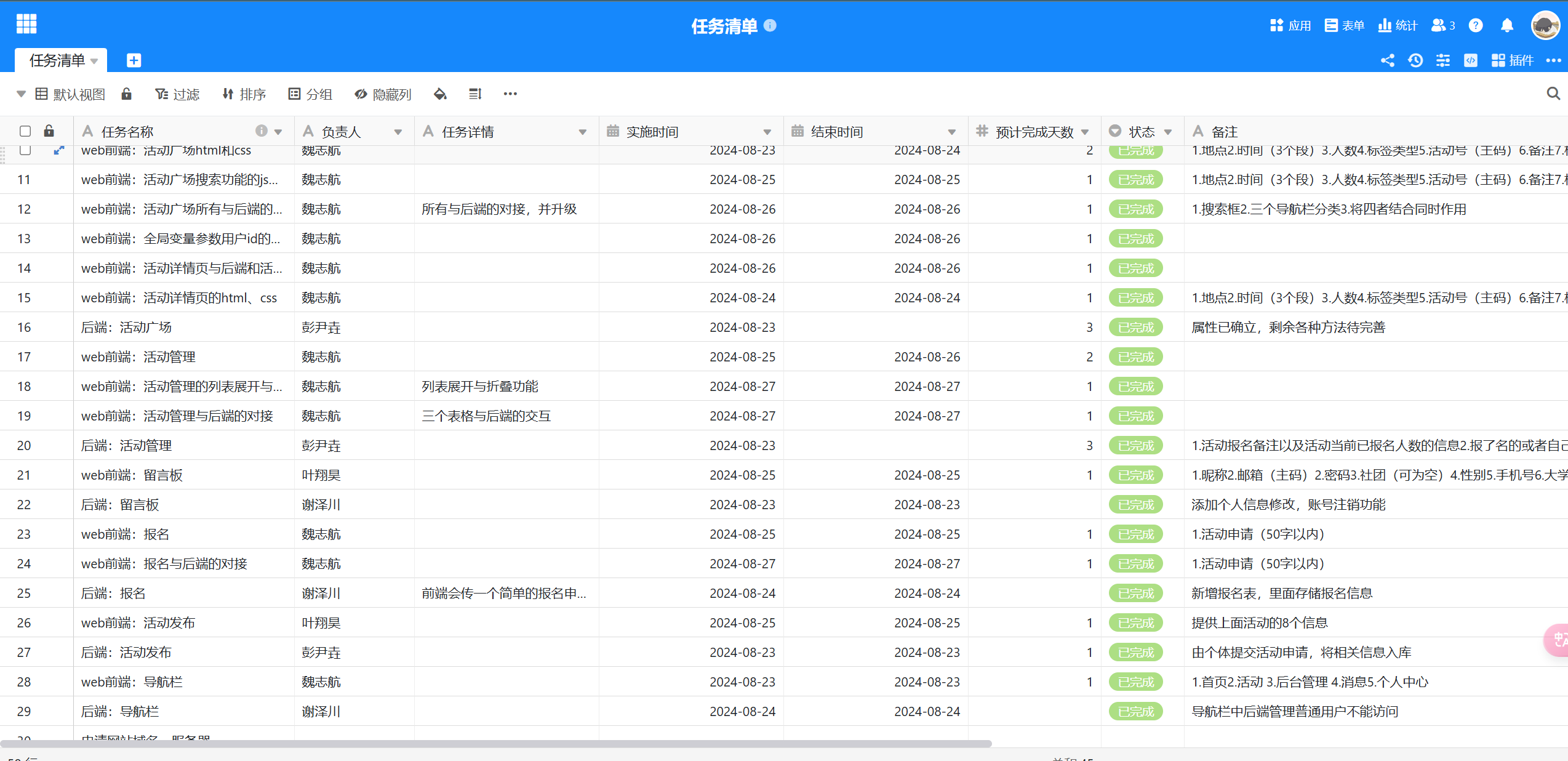
组长:魏志航

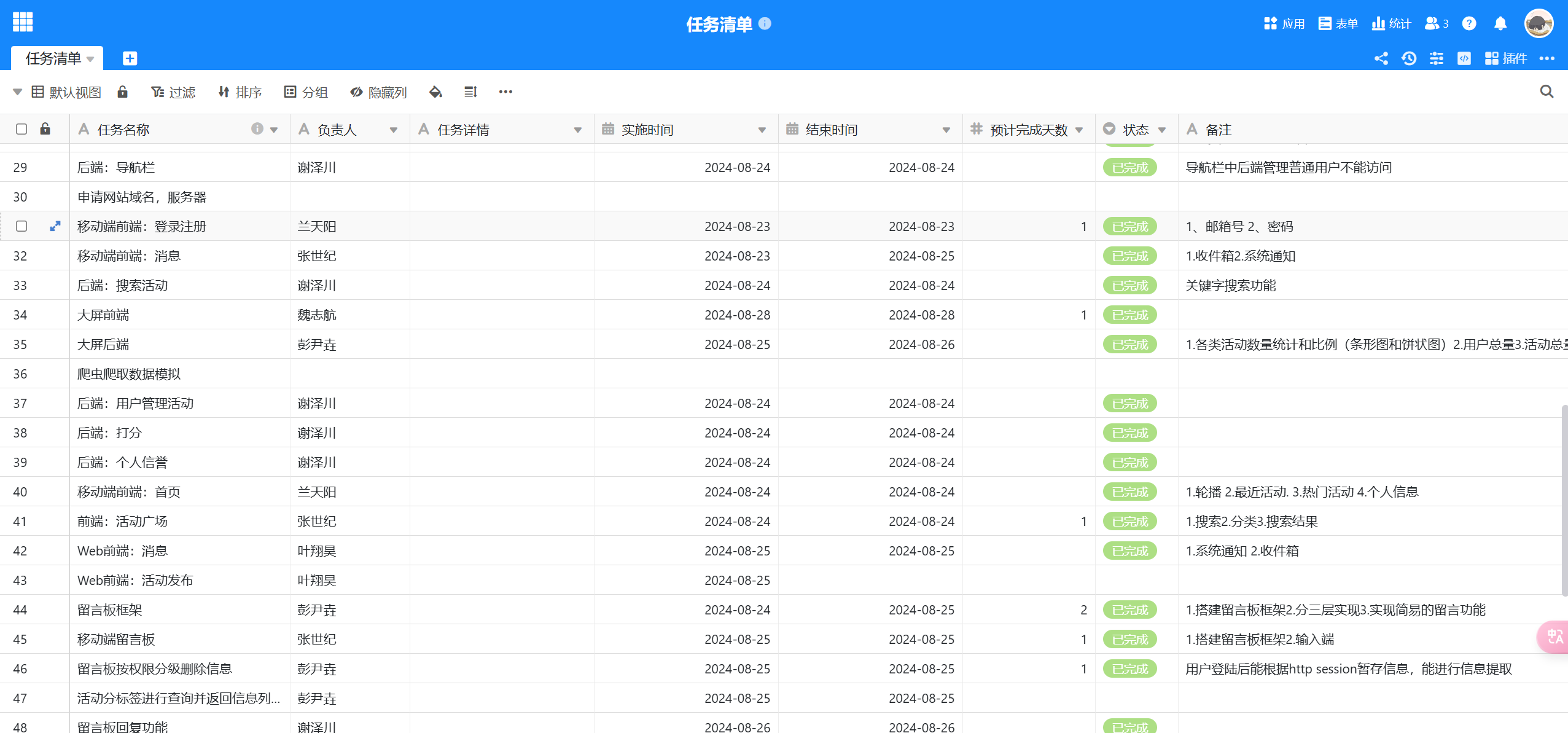
组员:兰天阳，彭尹垚，张世纪，叶翔昊，谢泽川

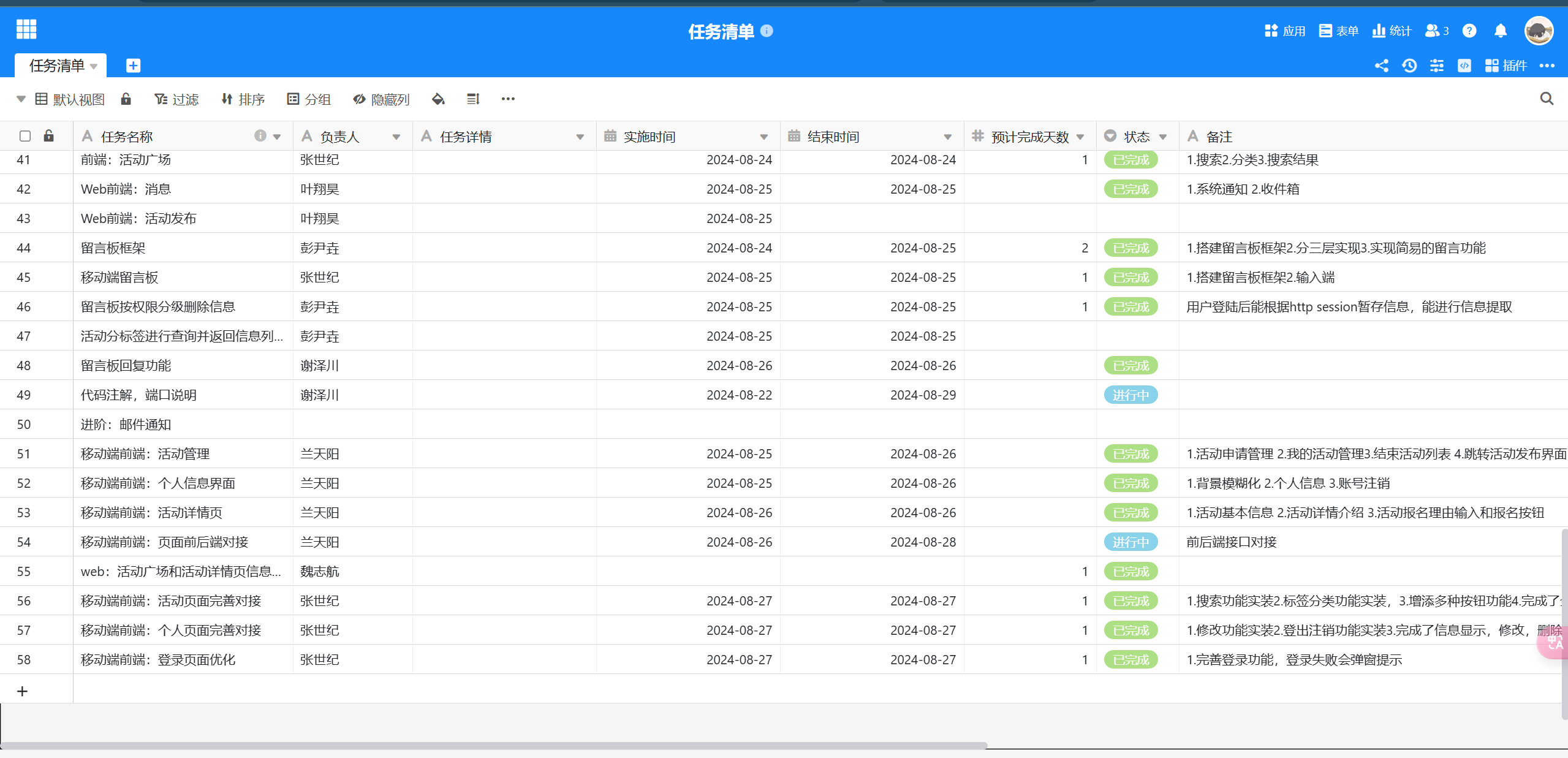
总体情况：

完成了前端与后端最后的对接，并对UI设计进行了优化，完善了一些bug。基本完成项目任务。

**魏志航**







**1.活动详情页和活动广场信息的完善，增加了显示当前活动已经报名的人数**

fetch(`http://192.168.1.43:8080/event/${activityId}/count`)

                        .then(response => {

                            if (!response.ok) {

                                throw new Error('网络响应错误');

                            }

                            return response.json();

                        })

                        .then(data => {

                            if (data.code === 200) {

                                const peoplenum = data.data; // 获取活动的具体信息

                                // 更新界面内容

                                document.getElementById('peopleNum').innerText =peoplenum;

                            }

                            else {

                                console.error('获取活动详细信息失败:', data.message);

                                console.error('获取活动详细信息失败:', data.code);

                                alert('加载活动详情失败，请重试。');

                            }

                        })

fetch(`http://192.168.1.43:8080/event/${activity.activityid}/count`)

                        .then(response => {

                            if (!response.ok) {

                                throw new Error('网络响应错误');

                            }

                            return response.json();

                        })

                        .then(countResult => {

                            if (countResult.code === 200) {

                                const peopleNow = countResult.data; // 获取当前人数

                                const participantCount = peopleNow; // 假设 peopleNow 代表当前参与人数

                                // 使用响应中的数据填充活动项

                                activityItem.innerHTML = `

                                    <div class="activity-img">

                                        <img src="${activity.profileimageurl || 'default\_image.png'}" alt="${activity.activityname}" />

                                    </div>

                                    <div class="activity-details">

                                        <h2>${activity.activityname}</h2>

                                        <p>发布人: ${activity.headusername}</p>

                                        <p>人数: ${participantCount}/${activity.maxceiling}人</p>

                                        <p>类型: ${activity.type}</p>

                                        <p>地点: ${activity.location || '未设置'}</p>

                                        <p>开始时间: ${formatDate(activity.timeproceed1 || '未设置')}</p>

                                    </div>

                                `;

                                activityList.appendChild(activityItem);

                            } else {

                                console.error('获取当前人数失败:', countResult.message);

                                // 如果没有返回一个有效的值，可以给出一个默认值

                                activityItem.innerHTML = `

                                    <div class="activity-img">

                                        <img src="${activity.profileimageurl || 'default\_image.png'}" alt="${activity.activityname}" />

                                    </div>

                                    <div class="activity-details">

                                        <h2>${activity.activityname}</h2>

                                        <p>发布人: ${activity.headusername}</p>

                                        <p>人数: 0/${activity.maxceiling}人</p> <!-- 默认人数为0，表示获取失败 -->

                                        <p>类型: ${activity.type}</p>

                                        <p>地点: ${activity.location || '未设置'}</p>

                                        <p>开始时间: ${formatDate(activity.timeproceed1 || '未设置')}</p>

                                    </div>

                                `;

                                activityList.appendChild(activityItem);

                            }

                        })

                        .catch(countError => {

                            console.error('获取人数时发生错误:', countError);

                            activityItem.innerHTML = `

                                <div class="activity-img">

                                    <img src="${activity.profileimageurl || 'default\_image.png'}" alt="${activity.activityname}" />

                                </div>

                                <div class="activity-details">

                                    <h2>${activity.activityname}</h2>

                                    <p>发布人: ${activity.headusername}</p>

                                    <p>人数: 出现错误</p>

                                    <p>类型: ${activity.type}</p>

                                    <p>地点: ${activity.location || '未设置'}</p>

                                    <p>开始时间: ${formatDate(activity.timeproceed1 || '未设置')}</p>

                                </div>

                            `;

                            activityList.appendChild(activityItem);

                        });

                });

            } else {

                // 如果没有活动数据

                activityList.innerHTML = '<p>当前没有活动报名。</p>';

            }

        } else {

            console.error('获取活动数据失败:', result.message);

            activityList.innerHTML = `<h2>暂无结果。</h2>`; // 显示错误消息

        }

    })

    .catch(error => {

        console.error('获取活动数据时发生错误:', error);

        activityList.innerHTML = '<p>网络错误，请稍后再试。</p>'; // 显示网络错误

    });

                }

                // 初始化加载所有活动

                loadActivities(); // 加载时显示所有活动

                activityList.addEventListener('click', function(event) {

                    // 检查事件目标是否是活动项

                    const activityItem = event.target.closest('.activity-item');

                    if (activityItem) {

                        const url = activityItem.getAttribute('data-url');

                        console.log("点击了activity项，目标URL是:", url);

                        if (url) {

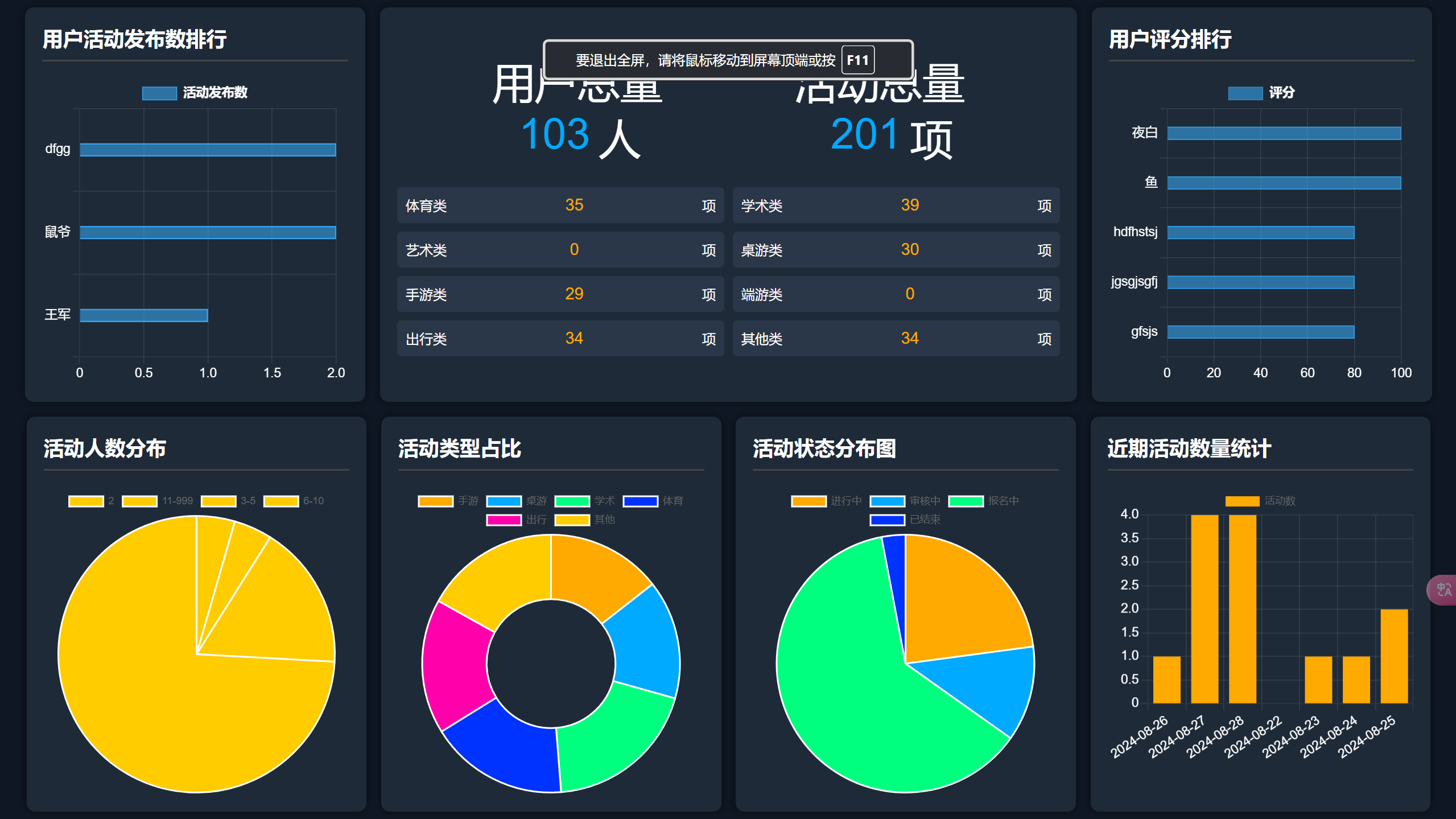
                            window.location.href = url; // 跳转

                        }

                    }

                });

**2.** **大屏前端及与后端的对接**



<!DOCTYPE html>

<html lang="zh">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>校园活动平台数据展示</title>

    <link rel="icon" type="image/png" href="../../images/logo.png" sizes="32x32">

    <link rel="stylesheet" href="BigScreen.css">

</head>

<body>

    <div class="dashboard">

        <div class="panel">

            <h2>用户活动发布数排行</h2>

            <canvas id="rankingChart"></canvas>

        </div>

        <div class="big">

            <div class="total-title">

                <span class="totaltitle">用户总量</span>

                <span class="totaltitle" >活动总量</span>

            </div>

            <div class="total">

                <div class="to">

                    <span class="total-people"></span>

                    <span style="margin-bottom: 10px;">人</span>

                </div>

                <div class="to"></span>

                    <span class="total-number" ></span>

                    <span>项</span>

                </div>

            </div>

            <div class="categories">

                <div>体育类<span class="category-number" id="sports" style="font-size: larger;"></span><span>项</span></div>

                <div>学术类<span class="category-number" id="science" style="font-size: larger;"></span><span>项</span></div>

                <div>艺术类<span class="category-number" id="art" style="font-size: larger;"></span><span>项</span></div>

                <div>桌游类<span class="category-number" id="table" style="font-size: larger;"></span><span>项</span></div>

                <div>手游类<span class="category-number" id="game" style="font-size: larger;"></span><span>项</span></div>

                <div>端游类<span class="category-number" id ="app" style="font-size: larger;"></span><span>项</span></div>

                <div>出行类<span class="category-number" id="travel" style="font-size: larger;"></span><span>项</span></div>

                <div>其他类<span class="category-number" id="other" style="font-size: larger;"></span><span>项</span></div>

            </div>

        </div>

        <div class="panel">

            <h2>用户评分排行</h2>

            <canvas id="typeStatsChart"></canvas>

        </div>

        <div class="panel">

            <h2>活动人数分布</h2>

            <canvas id="trendChart"></canvas>

        </div>

        <div class="panel">

            <h2>活动类型占比</h2>

            <canvas id="machineVsHumanChart"></canvas>

        </div>

        <div class="panel">

            <h2>活动状态分布图</h2>

            <canvas id="distributionChart"></canvas>

        </div>

        <div class="panel">

            <h2>近期活动数量统计</h2>

            <canvas id="typeStatsChart2"></canvas>

        </div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-datalabels"></script>

    <script src="BigScreen.js"></script>

</body>

</html>

body {

    margin: 0;

    font-family: Arial, sans-serif;

    background-color: hsl(217, 43%, 10%);

    color: #ffffff;

}

.dashboard {

    display: flex; /\* 使用 Grid 布局 \*/

    flex-direction: row;

    flex-wrap: wrap;

    justify-content: center; /\* 水平居中 \*/

    padding: 0; /\* 去掉内边距，以便完全填充 \*/

    height: 90vh; /\* 让 dashboard 填满整个视口高度 \*/

    width: 100vw; /\* 让 dashboard 填满整个视口宽度 \*/

}

.panel {

    background-color: #1e2a38;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);

    width: 21vw;

    height: 44vh;

    margin: 0.5%;

}

.big {

    background-color: #1e2a38;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);

    width: 45.5vw;

    height: 44vh;

    margin: 0.5% ;

    display: flex; /\* 使用 Flex 布局 \*/

    flex-direction: column; /\* 设置为纵向排列 \*/

    justify-content: center; /\* 在垂直方向上居中对齐 \*/

    text-align: center; /\* 文本居中 \*/

}

.panel h2 {

    margin: 0 0 20px;

    font-size: 24px;

    border-bottom: 2px solid #444;

    padding-bottom: 10px;

}

.total {

    font-size: 50px;

    display: flex;

    justify-content: center;

    text-align: center; /\* 文本居中 \*/

}

.to{

    margin-left: 7vw;

    margin-right: 6vw;

    display: flex;

    justify-content: center;

}

.total-title

{

    font-size: 50px;

    margin: 0;

}

.totaltitle

{

    margin: 0 70px;

}

.total-number,.total-people {

    color: #00aaff;

    margin-right: 10px;

}

.categories {

    display: flex;

    flex-wrap: wrap;

    gap: 10px;

    margin-top: 15px;

}

.categories div {

    background-color: #293748;

    padding: 10px;

    border-radius: 5px;

    flex: 1 1 30%;

    display: flex;

    justify-content: space-between;

    align-items: center;

}

.category-number {

    color: #ffaa00;

}

#typeStatsChart2 {

    width: 80%; /\* 设置宽度，可以调整 \*/

    height: 10px; /\* 设置高度 \*/

}

const options = {

    indexAxis: 'y', // 使条形图为垂直方向

    responsive: true,

    maintainAspectRatio: true, // 允许自定义宽高

    aspectRatio: 1, // 设置纵横比，调小这个值会使图表整体高度变小

    scales: {

        x: {

            beginAtZero: true, // X轴从零开始

            ticks: {

                color: 'white', // Y轴值的颜色设置为白色

                font: {

                    size: 15 // Y轴字体大小设置为20px

                }

            },

            grid: {

                color: 'rgba(255, 255, 255, 0.1)', // 网格线颜色可选

            }

        },

        y: {

            ticks: {

                color: 'white', // Y轴值的颜色设置为白色

                font: {

                    size: 15 // Y轴字体大小设置为20px

                }

            },

            grid: {

                color: 'rgba(255, 255, 255, 0.1)', // 网格线颜色可选

            },

        }

    },

    plugins: {

        legend: {

            display: true,

            position: 'top', // 图例位置

            labels: {

                font: {

                    size: 15, // 字体大小更改为 20px

                    family: "'Helvetica Neue', 'Helvetica', 'Arial', sans-serif", // 字体家族

                    weight: 'bold', // 字体粗细

                    style: 'normal', // 字体样式

                    lineHeight: 1.5, // 行高

                },

                color: 'white', // 标签颜色设置为白色

            }

        },

        datalabels: {

            color: 'white', // 数据标签颜色

            anchor: 'end',  // 标签位置，end 表示在条形的尾端

            align: 'end',   // 标签的对齐方式

            font: {

                size: 20, // 数据标签字体大小

                weight: 'bold'

            },

            formatter: (value) => {

                return value; // 显示的具体内容，这里是数值

            }

        }

    }

};

const options2 = {

    indexAxis: 'x', // 使条形图为垂直方向

    responsive: true,

    maintainAspectRatio: true, // 允许自定义宽高

    aspectRatio: 1.1, // 可选，设置纵横比，调小这个值会使图表整体高度变小

    scales: {

        x: {

            beginAtZero: true, // X轴从零开始

            ticks: {

                color: 'white', // Y轴值的颜色设置为白色

                font: {

                    size: 15 // Y轴字体大小设置为20px

                }

            },

            grid: {

                color: 'rgba(255, 255, 255, 0.1)', // 网格线颜色可选

            }

        },

        y: {

            ticks: {

                color: 'white', // Y轴值的颜色设置为白色

                font: {

                    size: 15 // Y轴字体大小设置为20px

                }

            },

            grid: {

                color: 'rgba(255, 255, 255, 0.1)', // 网格线颜色可选

            },

        }

    },

    plugins: {

        legend: {

            display: true,

            position: 'top' // 图例设置

        },

        formatter: (value) => {

            return value; // 显示的具体内容，这里是数值

        }

    },

};

// 分类数量

// 获取活动类型数据

async function fetchTypeData22() {

    try {

        const response = await fetch('http://192.168.1.43:8080/activities/typeCounts');

        if (!response.ok) throw new Error('网络响应错误'); // 错误处理

        const result = await response.json();

        if (result.code === 200) {

            // 更新网页中对应的类别项

            const typeCounts = result.data; // 获取数据对象

            // 将类型数据填充到相应的 span 元素

            document.getElementById('sports').textContent = typeCounts['体育'] || 0; // 填充体育类

            document.getElementById('science').textContent = typeCounts['学术'] || 0; // 填充学术类

            document.getElementById('art').textContent = typeCounts['艺术'] || 0; // 填充艺术类

            document.getElementById('table').textContent = typeCounts['桌游'] || 0; // 填充桌游类

            document.getElementById('game').textContent = typeCounts['手游'] || 0; // 填充手游类

            document.getElementById('app').textContent = typeCounts['端游'] || 0; // 填充端游类

            document.getElementById('travel').textContent = typeCounts['出行'] || 0; // 填充出行类

            document.getElementById('other').textContent = typeCounts['其他'] || 0; // 填充其他类

        } else {

            console.error('请求失败，返回信息:', result.message);

        }

    } catch (error) {

        console.error('获取数据出错:', error);

    }

}

// 用于渲染图表

function renderChart(id, type, data, options = {}) {

    new Chart(document.getElementById(id), {

        type: type,

        data: data,

        options: options

    });

}

function initialize() {

    refreshData(); // 初始化加载数据

    setInterval(refreshData, 3000); // 每3秒更新一次数据

}

window.onload = initialize;

 async function sum() {

    // 获取排名数据

    const rankingResponse = await fetch('http://192.168.1.43:8080/event/top5usernames');

    const rankingData = {

        labels: [], // 用户名

        datasets: [{

            label: '活动发布数',

            data: [], // 用户活动发布数

            backgroundColor: 'rgba(54, 162, 235, 0.6)',

            borderColor: 'rgba(54, 162, 235, 1)',

            borderWidth: 1,

            barThickness: 15,

        }]

    };

    // 处理排名数据

    if (rankingResponse.ok) {

        const rankingItems = await rankingResponse.json();

        rankingItems.forEach(item => {

            rankingData.labels.push(item.username);

            rankingData.datasets[0].data.push(item.count);

        });

    } else {

        console.error('获取排名数据出错');

    }

    // 渲染第一个图表

    renderChart('rankingChart', 'bar', rankingData,options);

    // 获取用户活动发布数排行数据

    const ranking2Response = await fetchRankingData(); // 假设这个函数是单独获取的

    const ranking2Data = {

        labels: ranking2Response.map(user => user.username), // 假设包含一个用户名

        datasets: [{

            label: '评分',

            data: ranking2Response.map(user => user.reputation), // 假设包含一个评分

            backgroundColor: 'rgba(54, 162, 235, 0.6)',

            borderColor: 'rgba(54, 162, 235, 1)',

            borderWidth: 0.5,

        barThickness: 15, // 设置每个条形的宽度（如：10）

        barPercentage: 0.4, // 将条形宽度设置成某种比例，0.4 较窄

        categoryPercentage: 0.3 // 类别间的相对宽度，0.8 保持一定的距离

        }]

    };

    // 渲染第二个图表

    renderChart('typeStatsChart', 'bar', ranking2Data,options);

    // 获取其他数据以及渲染其它图表

    const peopleDataResponse = await fetchPeopleData();

    const peopleData = {

        labels: Object.keys(peopleDataResponse),

        datasets: [{

            data: Object.values(peopleDataResponse),

            backgroundColor: '#ffcc00',

        }]

    };

    const typeDataResponse = await fetchTypeData();

    const typeData = {

        labels: Object.keys(typeDataResponse),

        datasets: [{

            data: Object.values(typeDataResponse),

            backgroundColor: ['#ffaa00', '#00aaff', '#00ff7f', '#0033ff', '#ff00aa', '#ffcc00', '#8080550', '#ffa440'],

        }]

    };

    const statusDataResponse = await fetchStatusData();

    const statusData = {

        labels: Object.keys(statusDataResponse),

        datasets: [{

            data: Object.values(statusDataResponse),

            backgroundColor: ['#ffaa00', '#00aaff', '#00ff7f', '#0033ff'],

        }]

    };

    const dateDataResponse = await fetchDateData();

    const dateData = {

        labels: Object.keys(dateDataResponse),

        datasets: [{

            label: '活动数',

            data: Object.values(dateDataResponse),

            backgroundColor: '#ffaa00',

        }]

    };

    // 渲染其它图表

    renderChart('trendChart', 'pie', peopleData);

    renderChart('machineVsHumanChart', 'doughnut', typeData);

    renderChart('distributionChart', 'pie', statusData);

    renderChart('typeStatsChart2', 'bar', dateData,options2);

};

// 获取用户活动发布数排行数据

async function fetchRankingData() {

    const response = await fetch('http://192.168.1.43:8080/auth/top5Reputation');

    const result = await response.json();

    return result.data; // 假设 result.data 为数组形式

}

// 获取活动人数分布数据

async function fetchPeopleData() {

    const response = await fetch('http://192.168.1.43:8080/activities/maxceilingcounts');

    const result = await response.json();

    return result.data; // 确保返回的数据为符合需要的格式

}

// 获取活动类型数据

async function fetchTypeData() {

    const response = await fetch('http://192.168.1.43:8080/activities/typeCounts');

    const result = await response.json();

    console.log(result.data);

    return result.data;

}

// 获取活动状态数据

async function fetchStatusData() {

    const response = await fetch('http://192.168.1.43:8080/activities/statusCounts');

    const result = await response.json();

    return result.data;

}

// 获取近期活动数量数据

async function fetchDateData() {

    const response = await fetch('http://192.168.1.43:8080/activities/countlast7days');

    const result = await response.json();

    return result;

}

//刷新

async function refreshData() {

    try {

        // 总用户

        const userResponse = await fetch('http://192.168.1.43:8080/auth/count');

        const userData = await userResponse.json();

        if (userData.code === 200) {

            document.querySelector('.total-people').textContent = userData.data;

        } else {

            console.error('请求失败，返回信息:', userData.message);

        }

        // 活动总量

        const activityResponse = await fetch('http://192.168.1.43:8080/activities/countActivity');

        const activityData = await activityResponse.json();

        if (activityData.code === 200) {

            document.querySelector('.total-number').textContent = activityData.data;

        } else {

            console.error('请求失败，返回信息:', activityData.message);

        }

        // 分类数量

        await fetchTypeData22(); // 你原本的函数可以直接调用

        await sum(); // 你原本的函数可以直接调用

    } catch (error) {

        console.error('网络请求出现错误:', error);

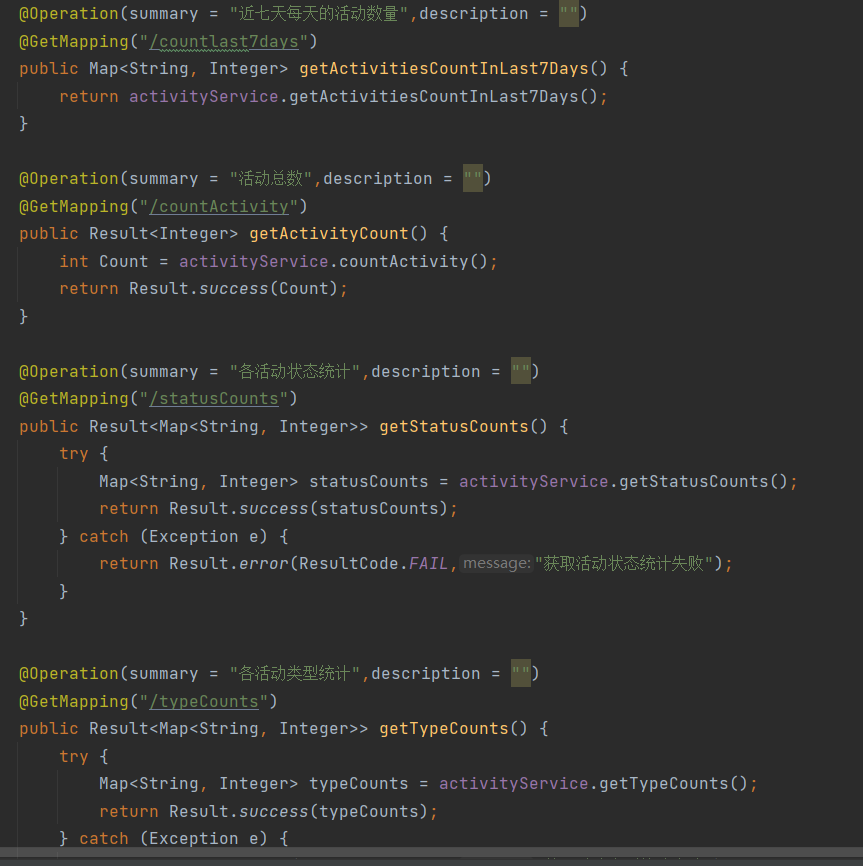
    }

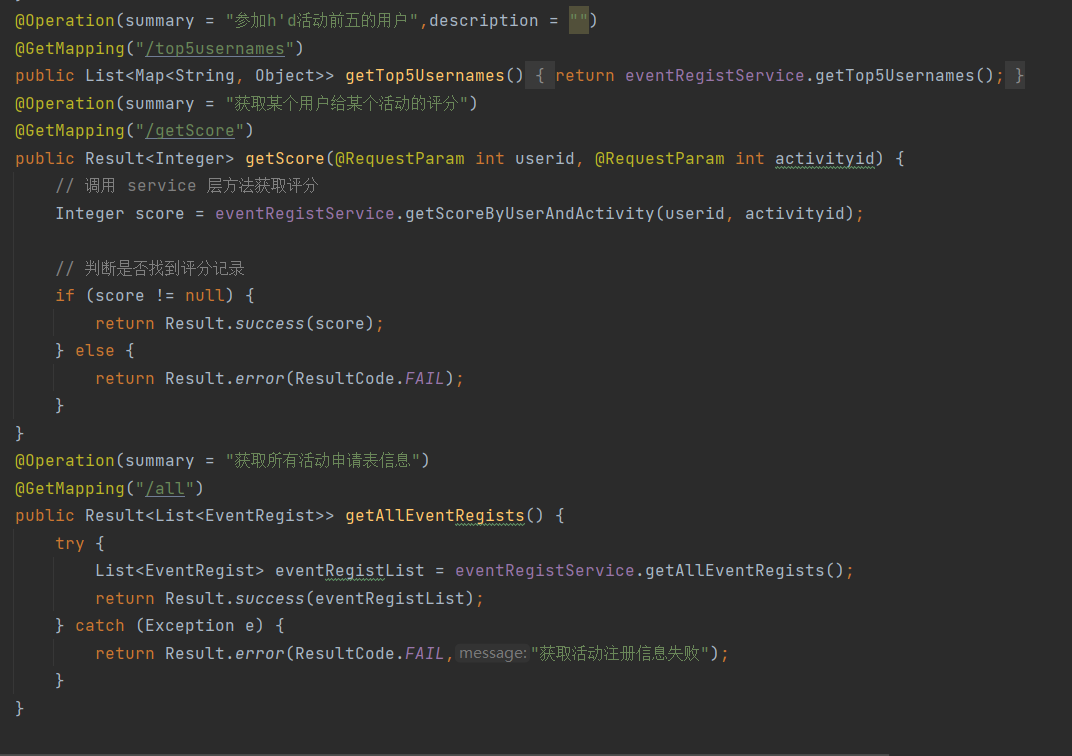
}

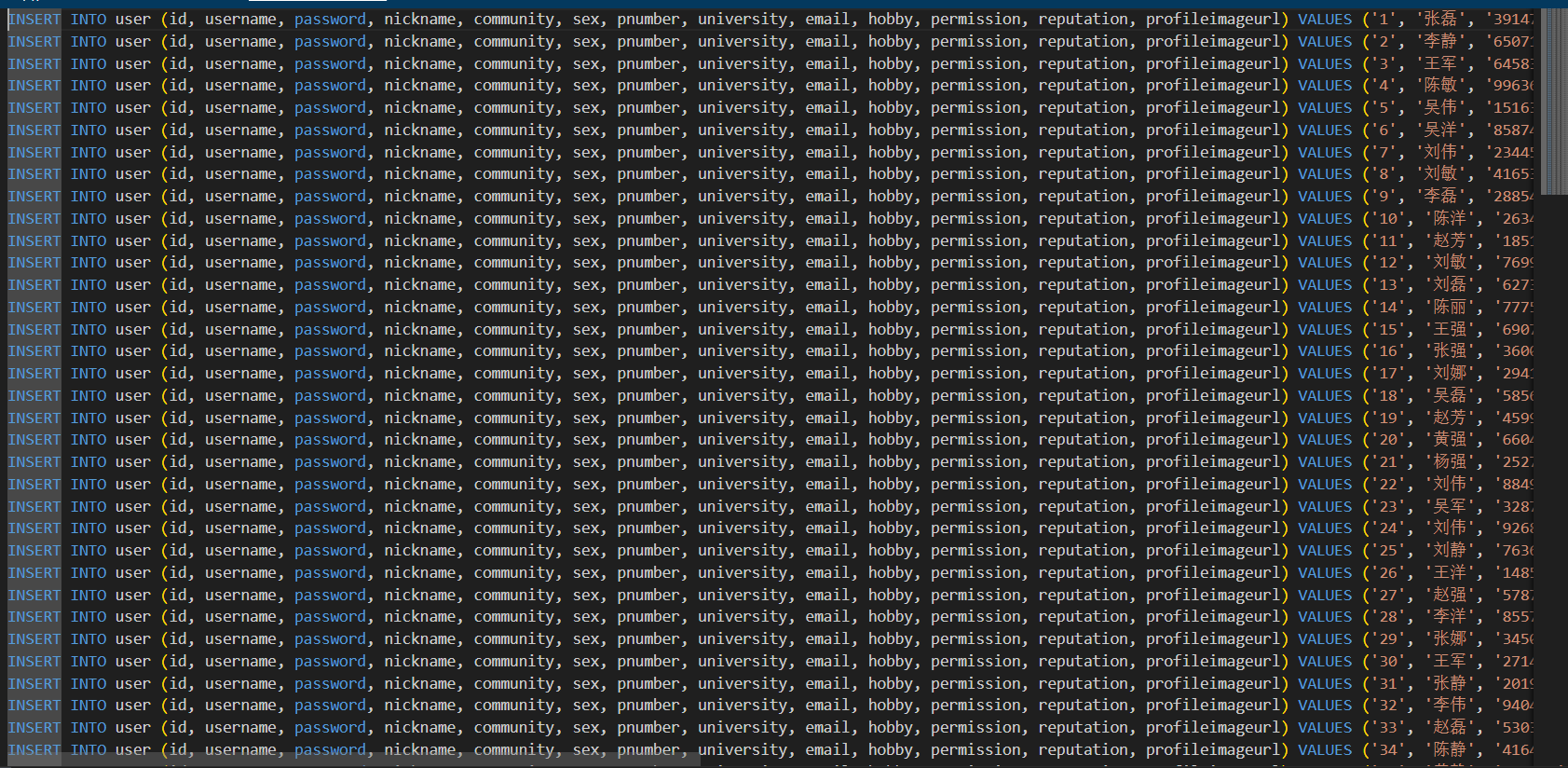
**彭尹垚**

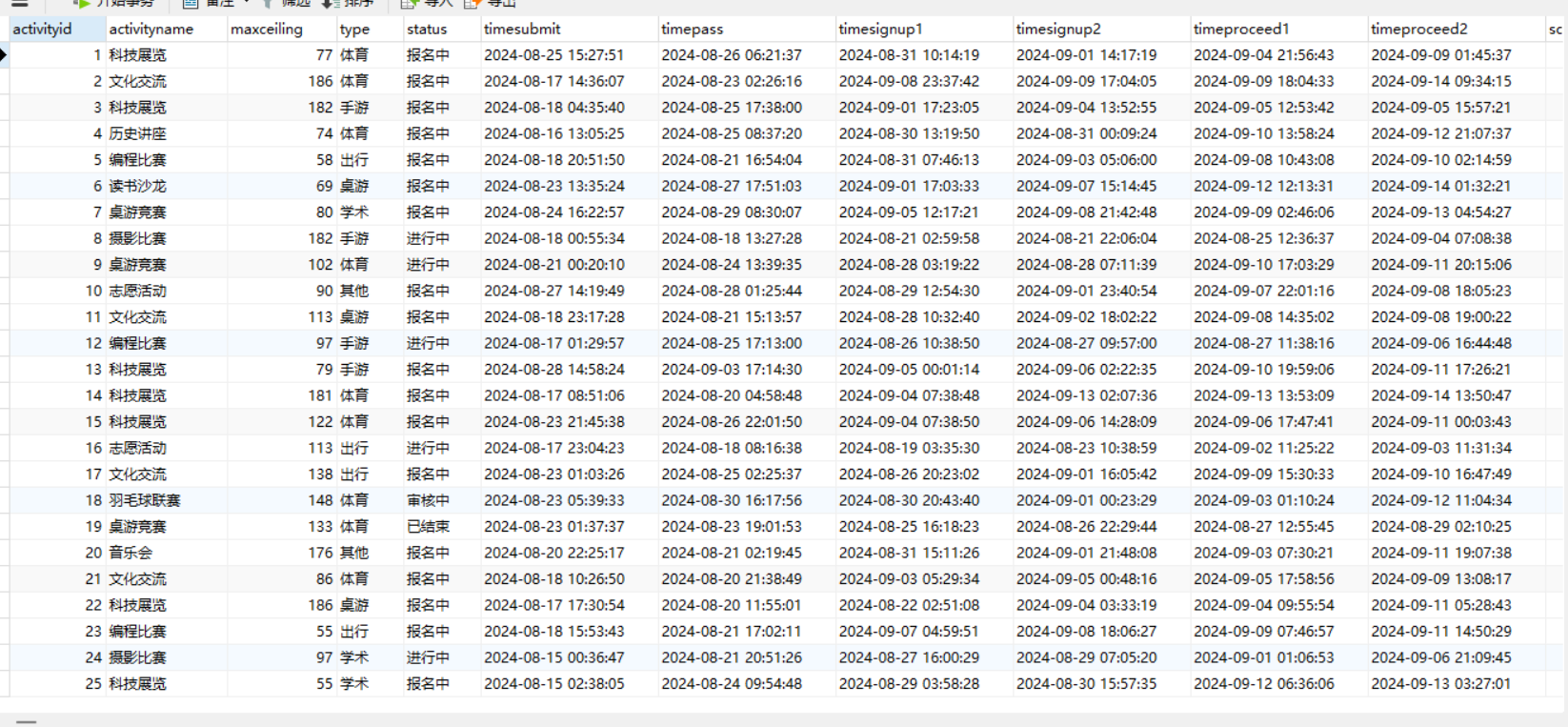
Java日志

今天进入了项目收尾阶段，大多数工作为与前端对接与debug，完成了大屏端的后端代码，对项目的一些功能进行了调整。对数据库的数据进行了爬取与调整，对后端部分的ppt进行制作。









**谢泽川**

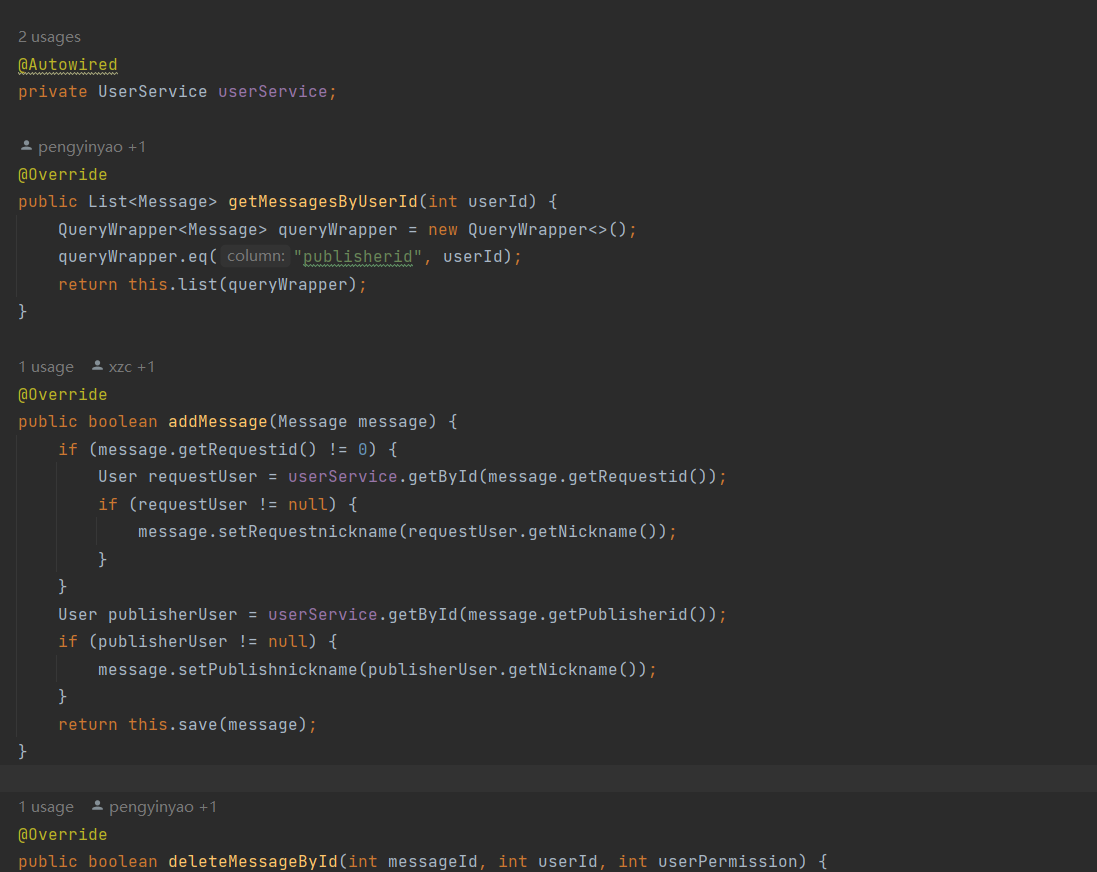
##### Java学习日志 - JAVASpringboot对校园活动的编写实现

##### 日期：2024年8月28日

##### 实现内容：

留言板的回复与发布功能：





该用户参与的活动（未结束的）的所有信息：



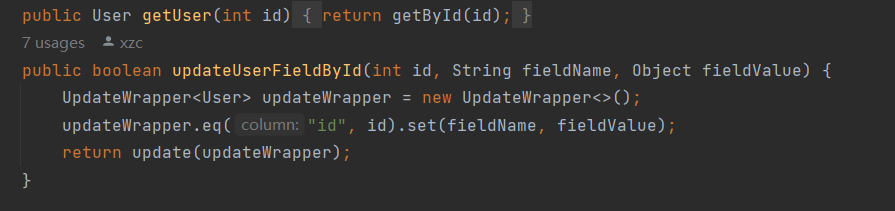


该用户参与的活动（已结束）的所有信息：





个人信息的修改：





**张世纪**

1. 完成了移动端活动页的端口对接，界面设计，实现了删除自己发布的活动，退出自己参加的活动的功能，实现了评分功能

2.全局优化调试

**叶翔昊**

8·27实训日志

总体情况：作为前端完成了所有功能的对接和网页的ui设计。

代码：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<style type="text/css">

\*{

margin: 0;

padding: 0;

}

body{

height: 100vh;

display: flex;

justify-content: center;

align-items: center;

}

.container{

position: absolute;

background-color: #90cbe3;

width:300px;

height: 500px;

border: 1px soid beige;

border-radius: 5%;

box-shadow: 10px 10px 15px 5px rgba(70, 70, 70, 0.6);

}

.content{

display: flex;

flex-direction: column;

padding: 100px 0 0px 0px;

}

.content.input{

margin: 0 auto 50px auto;

width: 300px;

height: 50px;

border: 1 0px solid rgba(187, 110, 225, 5);

}

.content.input:nth-child(2){

margin: 80px auto 50px auto;

width: 300px;

height: 50px;

border: 1px solid rgba(187, 110, 225, 5);

}

.content button{

margin: 40px auto 10px auto;

width: 300px;

height: 35px;

}

.content.span{

margin:40px auto 40px 110px;

width:300px;

height: 35px;

}

</style>

</head>

<body>

<div class="container">

<div class="content">

<input type="text" placeholder="用户名" />

<input type="password" placeholder="密码" />

<button type="button">登录</button>

<span>没有账号？<a href="#">去注册</a></span>

</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>用户登录</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">

<style>

html, body {

height: 100%;

margin: 0;

padding: 0;

font-family: Arial, sans-serif;

overflow: hidden;

}

body {

background: url('123/original.jpg') no-repeat center center fixed;

background-size: cover;

position: relative;

min-height: 100vh;

}

body::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.5);

z-index: -1;

}

.header {

text-align: center;

color: white;

margin-top: 50px;

font-size: 2em;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);

}

.container {

max-width: 400px;

margin: 100px auto;

padding: 20px;

background-color: rgba(255, 255, 255, 0.9);

border-radius: 15px; /\* 圆滑边缘 \*/

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /\* 阴影效果 \*/

position: relative;

width: 100%;

height: auto;

}

h2 {

text-align: center;

}

label {

display: block;

margin-top: 10px;

}

.password-container {

position: relative;

}

input[type="password"], input[type="text"] {

width: 100%;

padding: 8px;

margin-top: 5px;

box-sizing: border-box;

border-radius: 5px; /\* 输入框圆滑边缘 \*/

border: 1px solid #ccc; /\* 输入框边框 \*/

}

.toggle-password {

position: absolute;

right: 10px;

top: 50%;

transform: translateY(-50%);

cursor: pointer;

}

button {

width: 100%;

padding: 15px;

background-color: #28a745;

color: white;

border: none;

border-radius: 5px;

margin-top: 20px;

cursor: pointer;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

font-size: 16px;

font-weight: bold;

text-transform: uppercase;

}

button:hover {

background-color: #218838;

}

.error {

color: red;

margin-top: 10px;

text-align: center;

}

.switch-page {

text-align: center;

margin-top: 5px;

}

.switch-page a {

color: #007bff;

text-decoration: none;

}

.switch-page a:hover {

text-decoration: underline;

}

.quote { /\* 样式定义 \*/

text-align: center;

margin-top: 20px;

font-style: italic;

color: white;

text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.5);

}

</style>

</head>

<body>

<div class="header">学生活动发布系统</div>

<div class="container">

<h2>登录</h2>

<form id="loginForm">

<label for="loginUsername">账号：</label>

<input type="text" id="loginUsername" placeholder="用户名/邮箱" required>

<label for="loginPassword">密码：</label>

<div class="password-container">

<input type="password" id="loginPassword" placeholder="请输入密码" required>

<i class="fas fa-eye-slash toggle-password" onclick="togglePassword('loginPassword')"></i>

</div>

<button type="submit">登录</button>

<p class="error" id="loginError"></p>

</form>

<div class="switch-page">

<p>没有账户？<a href="register.html">点此注册</a></p>

</div>

</div>

<div class="quote" id="quote"></div> <!-- 显示名言的容器 -->

<script>

// 名言数组

const quotes = [

"「生活不只是眼前的苟且，还有诗和远方。」",

"「知之愈明，则行之愈笃;行之愈笃，则知之益明。」",

"「你必须参加生活中的每一场战斗，才能真正享受生活。」",

"「天空没有翅膀的痕迹，但鸟儿已经飞过。」",

"「行动是治愈恐惧的良药，而犹豫、拖延将不断滋养恐惧。」",

"「世界是一本大书，若不到处走走的话，看到的总是同一页的内容。」",

"「如果你不出去走走，你就会以为这就是世界。」",

];

// 随机选择名言并显示

function displayRandomQuote() {

const randomIndex = Math.floor(Math.random() \* quotes.length);

const quoteElement = document.getElementById('quote');

quoteElement.textContent = quotes[randomIndex];

}

// 页面加载时显示随机名言

window.onload = displayRandomQuote;

function togglePassword(inputId) {

const input = document.getElementById(inputId);

const icon = input.nextElementSibling;

if (input.type === "password") {

input.type = "text";

icon.classList.remove("fa-eye-slash");

icon.classList.add("fa-eye");

} else {

input.type = "password";

icon.classList.remove("fa-eye");

icon.classList.add("fa-eye-slash");

}

}

document.getElementById('loginForm').addEventListener('submit', async function (event) {

event.preventDefault();

const username = document.getElementById('loginUsername').value;

const password = document.getElementById('loginPassword').value;

try {

const response = await fetch('http://192.168.1.43:8080/auth/login', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded'

},

body: new URLSearchParams({

'usernameoremail': username,

'password': password

})

});

const result = await response.json();

console.log('Response Status:', response.status);

console.log('Response Data:', result);

if (response.ok && result.code === 200) {

// 保存用户信息到 localStorage

localStorage.setItem('user', JSON.stringify(result.data));

////////

const headid= result.data.id; // 这个值应该是从服务器获取的用户 ID

// 保存用户 ID 到 localStorage

localStorage.setItem('headid', headid);

console.log(headid);

///////

// 登录成功后跳转到主页

window.location.href = '../homepage/homepage.html';

} else {

document.getElementById('loginError').textContent = '账号或密码错误';

}

} catch (error) {

console.error('Error:', error); // 在控制台中记录错误

document.getElementById('loginError').textContent = '网络错误，请稍后重试';

}

});

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>用户注册</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">

<style>

html, body {

height: 100%;

margin: 0;

padding: 0;

font-family: Arial, sans-serif;

overflow: hidden;

}

body {

background: url('123/original.jpg') no-repeat center center fixed;

background-size: cover;

position: relative;

min-height: 100vh;

}

body::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.5);

z-index: -1;

}

.header {

font-family: "微软雅黑", sans-serif;

text-align: center;

color: white;

margin-top: 50px;

font-size: 2em;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);

}

.container {

max-width: 400px;

margin: 50px auto;

padding: 20px;

background-color: rgba(255, 255, 255, 0.9);

border-radius: 15px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

position: relative;

width: 100%;

height: auto;

}

h2 {

text-align: center;

}

label {

display: block;

margin-top: 10px;

}

.password-container {

position: relative;

}

input[type="password"], input[type="text"], input[type="email"] {

width: 100%;

padding: 8px;

margin-top: 5px;

box-sizing: border-box;

border-radius: 5px;

border: 1px solid #ccc;

}

.toggle-password {

position: absolute;

right: 10px;

top: 50%;

transform: translateY(-50%);

cursor: pointer;

}

.check-mark {

position: absolute;

right: 35px;

top: 50%;

transform: translateY(-50%);

color: green;

display: none;

}

.strength-bar-container {

display: none;

position: relative;

top: -20px; /\* 调整这个值来控制与输入框的距离 \*/

margin-bottom: -20px; /\* 确保与下一个元素有合适的间距 \*/

width: 50%;

margin-left: 55px;

}

.strength-bar {

display: flex;

height: 5px; /\* 调整整体高度 \*/

}

.strength-bar div {

flex: 1;

background-color: #ccc;

margin: 0 2px; /\* 为各个进度条添加间距 \*/

}

.strength-text {

font-size: 0.8em;

margin-top: 5px;

}

.error {

color: red;

margin-top: 10px;

text-align: center;

}

.switch-page {

text-align: center;

margin-top: 20px;

}

.switch-page a {

color: #007bff;

text-decoration: none;

}

.switch-page a:hover {

text-decoration: underline;

}

button {

width: 100%;

padding: 15px;

background-color: #28a745;

color: white;

border: none;

border-radius: 5px;

margin-top: 20px;

cursor: pointer;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

font-size: 16px;

font-weight: bold;

text-transform: uppercase;

}

button:hover {

background-color: #218838;

}

</style>

</head>

<body>

<div class="header">学生活动发布系统</div>

<div class="container">

<h2>注册</h2>

<form id="registerForm">

<label for="registerUsername">用户名：</label>

<input type="text" id="registerUsername" name="username" placeholder="请输入用户名" required>

<label for="registerEmail">邮箱：</label>

<input type="email" id="registerEmail" name="email" placeholder="请输入邮箱" required>

<label for="registerPassword">密码：</label>

<div class="strength-bar-container" id="strengthBarContainer">

<div class="strength-bar">

<div id="bar1"></div>

<div id="bar2"></div>

<div id="bar3"></div>

<div id="bar4"></div>

<div id="bar5"></div>

</div>

<div class="strength-text" id="strengthText">密码强度：非常弱</div>

</div>

<div class="password-container">

<input type="password" id="registerPassword" name="password" placeholder="请输入密码" required>

<i class="fas fa-eye-slash toggle-password" onclick="togglePassword('registerPassword')"></i>

<i class="fas fa-check check-mark" id="passwordMatchCheck"></i>

</div>

<label for="confirmPassword">确认密码：</label>

<div class="password-container">

<input type="password" id="confirmPassword" name="confirmPassword" placeholder="请再次输入密码" required>

<i class="fas fa-eye-slash toggle-password" onclick="togglePassword('confirmPassword')"></i>

<i class="fas fa-check check-mark" id="confirmMatchCheck"></i>

</div>

<button type="submit">注册</button>

<p class="error" id="registerError"></p>

</form>

<div class="switch-page">

<p>已有账户？<a href="login.html">点此登录</a></p>

</div>

</div>

<script>

function togglePassword(inputId) {

const input = document.getElementById(inputId);

const icon = input.nextElementSibling;

if (input.type === "password") {

input.type = "text";

icon.classList.remove("fa-eye-slash");

icon.classList.add("fa-eye");

} else {

input.type = "password";

icon.classList.remove("fa-eye");

icon.classList.add("fa-eye-slash");

}

}

function checkPasswordStrength(password) {

const strengthBarContainer = document.getElementById('strengthBarContainer');

const strengthBar = [

document.getElementById('bar1'),

document.getElementById('bar2'),

document.getElementById('bar3'),

document.getElementById('bar4'),

document.getElementById('bar5')

];

const strengthText = document.getElementById('strengthText');

if (password) {

strengthBarContainer.style.display = 'block';

} else {

strengthBarContainer.style.display = 'none';

return;

}

let score = 0;

// 密码长度

if (password.length <= 4) {

score += 5;

} else if (password.length <= 7) {

score += 10;

} else {

score += 25;

}

// 字母

const hasLower = /[a-z]/.test(password);

const hasUpper = /[A-Z]/.test(password);

if (hasLower || hasUpper) {

score += hasLower && hasUpper ? 20 : 10;

}

// 数字

const numMatches = (password.match(/\d/g) || []).length;

if (numMatches > 0) {

score += numMatches === 1 ? 10 : 20;

}

// 符号

const symbolMatches = (password.match(/[^a-zA-Z0-9]/g) || []).length;

if (symbolMatches > 0) {

score += symbolMatches === 1 ? 10 : 25;

}

// 奖励

if (hasLower || hasUpper) {

if (numMatches > 0) {

if (symbolMatches > 0) {

score += hasLower && hasUpper ? 5 : 3;

} else {

score += 2;

}

}

}

// 更新进度条颜色

strengthBar.forEach(bar => bar.style.backgroundColor = '#ccc');

if (score >= 90) {

strengthText.textContent = '密码强度：非常安全';

strengthBar.forEach(bar => bar.style.backgroundColor = 'purple');

} else if (score >= 80) {

strengthText.textContent = '密码强度：安全';

strengthBar.slice(0, 4).forEach(bar => bar.style.backgroundColor = 'purple');

} else if (score >= 70) {

strengthText.textContent = '密码强度：非常强';

strengthBar.slice(0, 3).forEach(bar => bar.style.backgroundColor = 'red');

} else if (score >= 60) {

strengthText.textContent = '密码强度：强';

strengthBar.slice(0, 3).forEach(bar => bar.style.backgroundColor = 'orange');

} else if (score >= 50) {

strengthText.textContent = '密码强度：一般';

strengthBar.slice(0, 2).forEach(bar => bar.style.backgroundColor = 'green');

} else if (score >= 25) {

strengthText.textContent = '密码强度：弱';

strengthBar.slice(0, 1).forEach(bar => bar.style.backgroundColor = 'green');

} else {

strengthText.textContent = '密码强度：非常弱';

}

}

function checkPasswordsMatch() {

const password = document.getElementById('registerPassword').value;

const confirmPassword = document.getElementById('confirmPassword').value;

const passwordMatchCheck = document.getElementById('passwordMatchCheck');

const confirmMatchCheck = document.getElementById('confirmMatchCheck');

if (password && confirmPassword && password === confirmPassword) {

passwordMatchCheck.style.display = 'inline';

confirmMatchCheck.style.display = 'inline';

} else {

passwordMatchCheck.style.display = 'none';

confirmMatchCheck.style.display = 'none';

}

}

document.getElementById('registerPassword').addEventListener('input', function () {

const password = this.value;

checkPasswordStrength(password);

checkPasswordsMatch();

});

document.getElementById('confirmPassword').addEventListener('input', checkPasswordsMatch);

document.getElementById('registerForm').addEventListener('submit', async function(event) {

event.preventDefault(); // 防止表单默认提交

const username = document.getElementById('registerUsername').value;

const email = document.getElementById('registerEmail').value;

const password = document.getElementById('registerPassword').value;

const confirmPassword = document.getElementById('confirmPassword').value;

if (password !== confirmPassword) {

document.getElementById('registerError').textContent = '密码和确认密码不一致';

return;

}

const userData = {

username: username,

password: password,

email: email

};

try {

const response = await fetch('http://192.168.1.43:8080/auth/register', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({

id: 0,

username: userData.username,

password: userData.password,

email: userData.email,

nickname: "",

community: "",

sex: "",

pnumber: 0,

university: "",

hobby: "",

permission: 0,

reputation: 0,

profileimageurl: ""

})

});

const result = await response.json();

if (response.ok&&result.code === 200) {

alert('注册成功！');

window.location.href = 'login.html'; // 跳转到登录页面

} else if (response.status === 400) {

document.getElementById('registerError').textContent = '用户名已存在';

} else {

document.getElementById('registerError').textContent = '用户名已存在';

}

} catch (error) {

console.error('Error:', error);

document.getElementById('registerError').textContent = '注册失败，请稍后再试。';

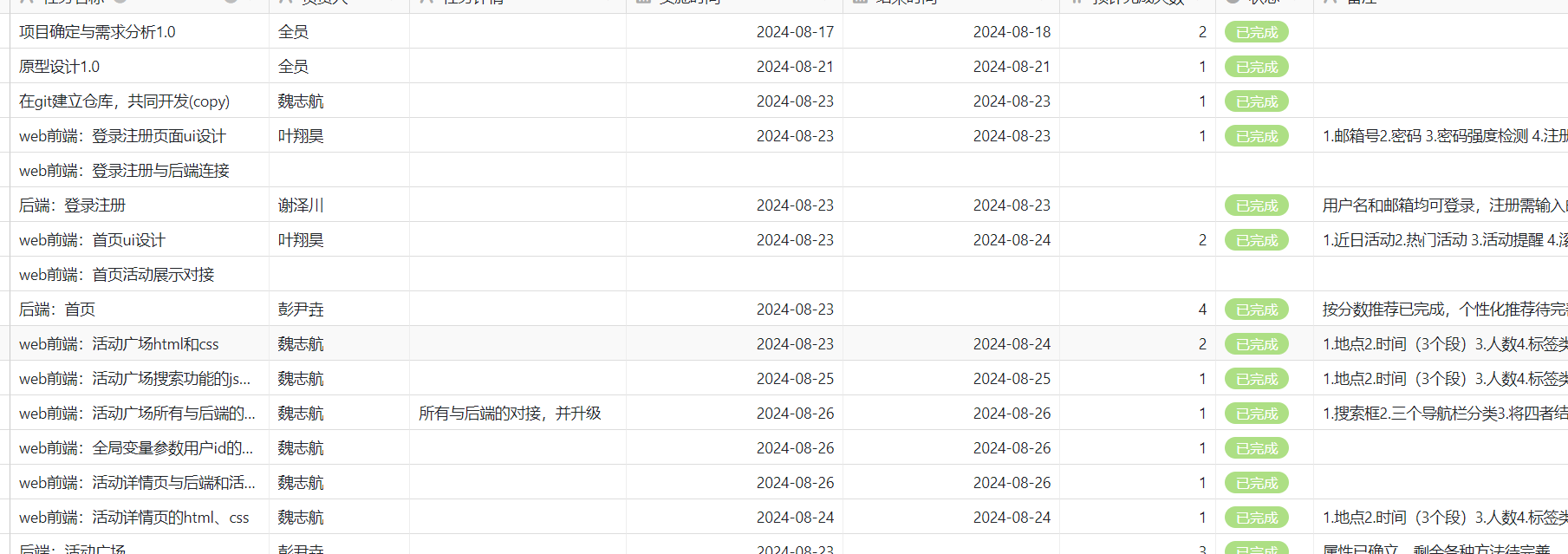
}

});

</script>

</body>

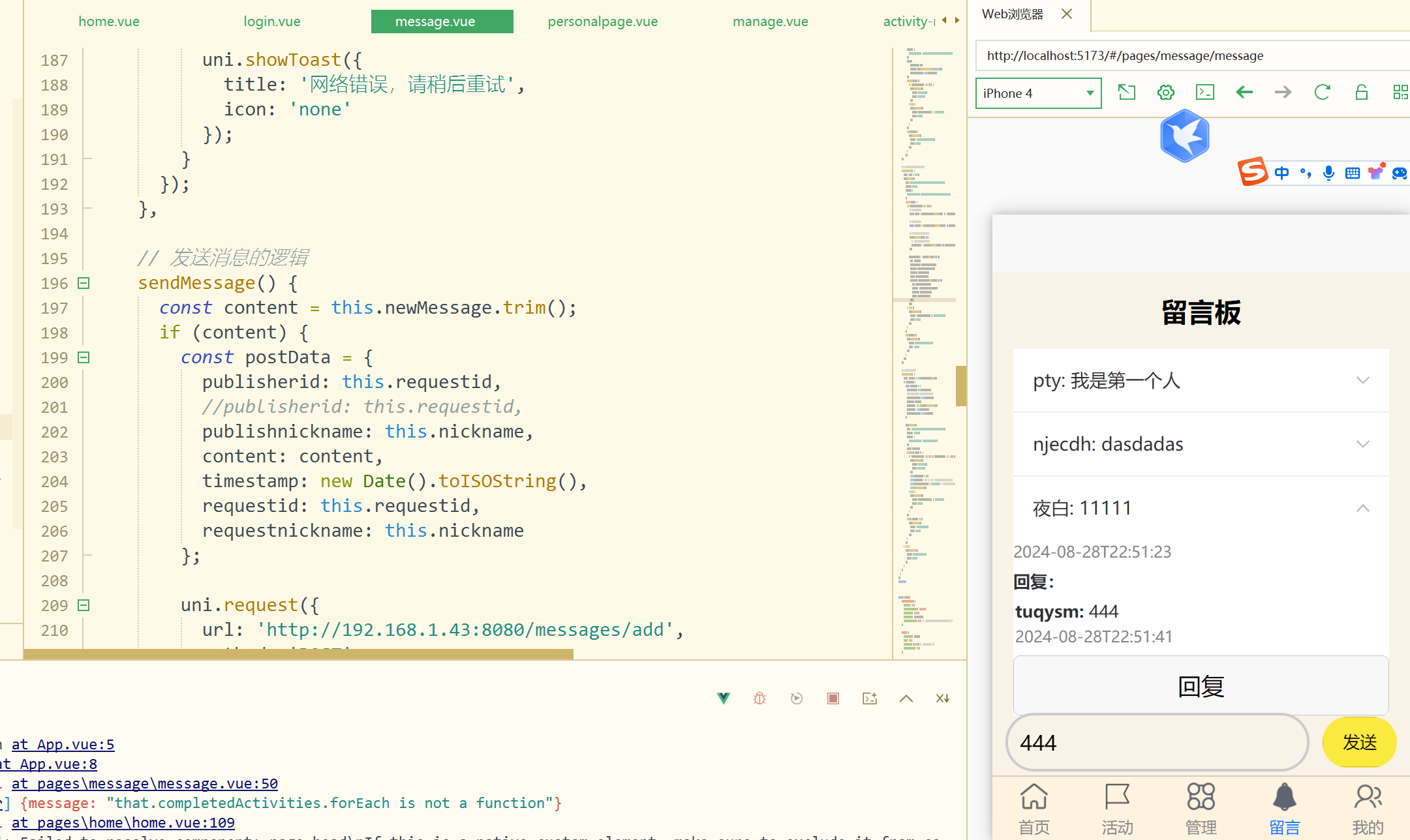
</html>



**兰天阳**

2024.8.28

完成移动端对接并测试，但发布框功能仍有待完善

<template>

<div class="message-board">

<!-- 留言板标题 -->

<h2 class="title">留言板</h2>

<!-- 留言列表 -->

<uni-collapse accordion="true">

<uni-collapse-item class="uni-collapse"

v-for="message in messages"

:key="message.publisherid"

:title="message.author + ': ' + message.content">

<p class="timestamp">{{ message.time }}</p>

<!-- 如果当前用户是发布者，则显示撤回按钮 -->

<button

v-if="message.publisherid === userid"

@click="retractMessage(message.id, message.publisherid)"

class="retract-btn">撤回</button>

<!-- 回复列表 -->

<div v-if="message.replies.length > 0" class="reply-list">

<p><strong>回复：</strong></p>

<div v-for="reply in message.replies" :key="reply.id" class="reply-item">

<p><strong>{{ reply.author }}:</strong> {{ reply.content }}</p>

<p class="timestamp">{{ reply.time }}</p>

</div>

</div>

<!-- 回复按钮 -->

<button @click="prepareReply(message)" class="requestbtn">回复</button>

</uni-collapse-item>

</uni-collapse>

<!-- 留言输入框 -->

<div class="message-input">

<input

v-model="newMessage"

type="text"

:placeholder="inputPlaceholder"

@keydown.enter="sendMessage"

/>

<button @click="sendMessage" class="send-button">发送</button>

</div>

</div>

</template>

<script>

export default {

onLoad() {

const userId = uni.getStorageSync('userid');

console.log(userId);

if (userId) {

this.userid = userId;

this.getUserInfo();

this.getAllPosts();

} else {

console.log('用户ID未找到');

}

},

data() {

return {

userid: uni.getStorageSync('userid'), // 用户ID

nickname: '', // 用户昵称

newMessage: '', // 新的留言内容

messages: [], // 留言列表

requestid: 0, // 如果是回复，则为回复的目标发布者的publisherid

inputPlaceholder: '输入...',

permission: 0,

publisherid: uni.getStorageSync('userid'),

};

},

methods: {

// 准备回复的逻辑

prepareReply(message) {

message.requestnickname = this.nickname;

this.requestid = message.publisherid; // 设置为所回复帖子的发布者ID

this.inputPlaceholder = `回复 ${message.author}`; // 修改输入框提示文字

},

// 获取用户信息

getUserInfo() {

const that = this;

uni.request({

url: 'http://192.168.1.43:8080/auth/showuser/' + this.userid,

method: 'GET',

header:{

'Content-Type': 'application/x-www-form-urlencoded'

},

success(res) {

if (res.statusCode === 200) {

that.nickname = res.data.data.nickname;

that.permission = res.data.data.permission;

} else {

uni.showToast({

title: res.data.message || '获取信息错误',

icon: 'none'

});

}

},

fail(error) {

uni.showToast({

title: '网络错误，请稍后重试',

icon: 'none'

});

}

});

},

retractMessage(id, publisherid){

console.log(id);

uni.request({

url: 'http://192.168.1.43:8080/messages/delete/' + id,

method: 'DELETE',

header: {

'Content-Type': 'application/x-www-form-urlencoded'

},

data:{

messageId: id,

userId: uni.getStorageSync('userid'),

userPermission: this.permission

},

success(res) {

if (res.statusCode === 200) {

uni.showToast({

title: '删除成功',

icon: 'success'

});

} else {

uni.showToast({

title: res.data.message || '删除失败',

icon: 'none'

});

}

},

fail(error) {

uni.showToast({

title: '网络错误，请稍后重试',

icon: 'none'

});

}

});

},

// 获取所有帖子及其对应的回复

getAllPosts() {

const that = this;

uni.request({

url: 'http://192.168.1.43:8080/messages/list',

method: 'GET',

header: {

'Content-Type': 'application/x-www-form-urlencoded'

},

success(res) {

if (res.statusCode === 200) {

// 获取所有帖子

const posts = res.data.data.filter(post => post.requestid === 0 && post.publisherid !== 0);

// 获取所有回复

const replies = res.data.data.filter(reply => reply.requestid !== 0 && reply.publisherid !== 0);

// 为每个帖子找出它的回复

posts.forEach(post => {

// 将回复归类到该帖子中

post.replies = replies.filter(reply => reply.publisherid === post.publisherid);

});

that.messages = posts.map(post => ({

id: post.id,

publisherid: post.publisherid,

author: post.publishnickname,

content: post.content,

time: post.timestamp,

replies: post.replies.map(reply => ({

id: reply.publisherid,

author: reply.requestnickname,

content: reply.content,

time: reply.timestamp

}))

}));

} else {

uni.showToast({

title: res.data.message || '获取帖子失败',

icon: 'none'

});

}

},

fail(error) {

uni.showToast({

title: '网络错误，请稍后重试',

icon: 'none'

});

}

});

},

// 发送消息的逻辑

sendMessage() {

const content = this.newMessage.trim();

if (content) {

const postData = {

publisherid: this.requestid,

//publisherid: this.requestid,

publishnickname: this.nickname,

content: content,

timestamp: new Date().toISOString(),

requestid: this.requestid,

requestnickname: this.nickname

};

uni.request({

url: 'http://192.168.1.43:8080/messages/add',

method: 'POST',

header: {

'Content-Type': 'application/json'

},

data: postData,

success: (res) => {

if (res.statusCode === 200 && res.data.code === 200) {

uni.showToast({

title: '发布成功',

icon: 'success'

});

this.newMessage = '';

this.requestid = 0; // 重置 requestid，表示下次发帖

this.inputPlaceholder = '输入...'; // 重置输入框提示文字

this.getAllPosts();

} else {

uni.showToast({

title: res.data.message || '发布失败',

icon: 'none'

});

}

},

fail: (error) => {

uni.showToast({

title: '发布请求失败',

icon: 'none'

});

}

});

} else {

uni.showToast({

title: '请填写帖子内容',

icon: 'none'

});

}

}

}

};

</script>

<style scoped>

.message-board {

padding: 5vw;

background-color: #f9f2e7;

min-height: 100vh;

box-sizing: border-box;

padding-bottom: 30vh; /\* 给底部留出空间，防止被输入框覆盖 \*/

}

.title {

text-align: center;

left: 40%;

font-size: 1.5rem; /\* 相对字体大小 \*/

margin-bottom: 5vw;

}

.message-list {

margin-bottom: 5vw;

}

.message-item {

background-color: #fff;

padding: 4vw;

margin-bottom: 3vw;

border-radius: 2vw;

box-shadow: 0 2vw 4vw rgba(0, 0, 0, 0.1);

}

.timestamp {

font-size: 3vw; /\* 相对字体大小 \*/

color: #888;

margin-bottom: 2vw;

}

.message-input {

position: fixed;

bottom: 9vh;

left: 0;

right: 0;

display: flex;

align-items: center;

padding: 3vw;

background-color: transparent;

}

.message-input input {

flex: 1;

padding: 3vw;

border: 2px solid #ccc;

border-radius: 10vw;

background-color: #f9f2e7;

}

.send-button {

background-color: #faea3f;

padding: 1vw 5vw;

border-radius: 10vw;

margin-left: 3vw;

cursor: pointer;

font-size: 4vw;

}

.retract-btn.requestbtn{

width: 60%;

border-radius: 1.5rem;

background-color: #faea3f;

}

.reply-item{

padding-top: 0.3rem;

}

.reply-item p {

padding: 0.1rem;

}

</style>